N90-25525

Knowledge Representation to Support Reasoning Based on Multiple Models

April Gillam, Jorge P. Seidel
Computer Science Laboratory
The Aerospace Corporation
Los Angeles, CA 90009
and Alice C. Parker
Dept. of Electrical Engineering - Systems
University of Southern California

This work was supported by the
Aerospace Sponsored Research Program.

1 ABSTRACT

Model Based Reasoning is a powerful tool used to design and analyze systems, which are often composed of numerous interactive, interrelated subsystems. Models of the subsystems are written independently and may be used together while they are still under development. Thus the models are not static. They evolve as information becomes obsolete, as improved artifact descriptions are developed, and as system capabilities change. We are using three methods to support knowledge/data base growth, to track the model evolution, and to handle knowledge from diverse domains. First, the representation methodology is based on having pools, or types, of knowledge from which each model is constructed. In addition information is explicit. This includes the interactions between components, the description of the artifact structure, and the constraints and limitations of the models. The third principle we have followed is the separation of the data and knowledge from the inferencing and equation solving mechanisms. This methodology is used in two distinct knowledge-based systems: one for the design of space systems and another for the synthesis of VLSI circuits. It has facilitated the growth and evolution of our models, made accountability of results explicit, and provided credibility for the user community. These capabilities have been implemented and are being used in actual design projects.

2 INTRODUCTION

Model Based Reasoning is a powerful tool used to design and analyze systems composed of numerous interactive, interrelated subsystems. The development of these complex systems requires the following basic steps: specification, modeling, and physical implementation. The specification itself covers three description levels: behavior, logic and structure. The ease with which knowledge is utilized depends on the representation scheme of both the knowledge and the design data. For example, a query to retrieve information might be a one line statement or might entail making several queries, each of which depends on interpreting the results of the previous query. Finding all possible modules active during a particular time sequence might require searching the entire design or might be a single table-lookup query. This paper describes a knowledge representation scheme that can be used at both the specification and model levels. Reasoning from models may cut across any of the levels mentioned and combine information of different models.

One difficulty in knowledge representation is caused by the dynamic nature of the information. The models evolve with the addition and deletion of information, the enhancement of capabilities in the artifact being designed, and the development of better model concepts. Also, the relationships between the models and the knowledge associated with them may evolve.

In this paper, we will address how our system of knowledge representation supports the following aspects of modelling:

- Synthesis of the model from behavioral specification
- Interactions between models
- Evolving models
- Representation of knowledge from diverse domains

The knowledge representation must be capable of expressing the behavior and structure of the system or artifact being designed. These may be described in terms of models which in turn are composed of many different kinds of information, such as equations, constraints, and algorithms. As this information grows and changes it is important to track the evolution. Work done early in a project frequently cannot be used when one returns to that early design, made 6 months ago. Without the models upon which the design decisions were made the work may need to be duplicated using the current models. It would be a lot simpler if the state of the models for each design were kept and the changes were easily accessible. Few people remember whether there were minor or major changes and when they've been made. This is especially true when dealing with a complex system which many designers.

While this form of Model Based Reasoning (MBR) will support many distinct design domains, disparate domains will be used here to illustrate the concepts. The first involves a knowledge-based system, VEHICLES [?] and [?], developed at the Aerospace Corp., that supports the conceptual design of space systems. There are several models for the spacecraft subsystems and their environments (e.g. payload, communications, launch, thermal, etc.). Integrating multiple models that are developed by different people who have focused on different aspects of the design, often at different levels as well, is quite challenging. The knowledge representation scheme we have developed makes the problem tenable.

The second domain involves high level synthesis of VLSI circuits from behavioral level specifications. The knowledge model used in KNOWledge MANager (KNOWMAN) (part of the Advanced Design AutoMation (ADAM) system developed at the University of Southern California [?] and [?] and [?]) must support many types of knowledge used in the various stages of automated synthesis of VLSI circuits from behavioral descriptions. KNOWMAN consists of a representation methodology for handling the design knowledge necessary for VLSI Synthesis, the implementation of the knowledge representation schema using an object oriented database model, and a set of Prolog expert systems that utilize the data in the database.

The important aspect of this paper is not the specific implementations, but the general application of our proposed methodology to these and other implementations. In Section 2, the basic scheme for knowledge representation will be outlined, and the major concepts associated with the proposed methodology will be stated. Section 3 will present an overview of the knowledge classification structure used in our design systems. Section 4 will summarize and present the status of our work and discuss related problems we are currently working on.

3 REPRESENTATION SCHEME

Our scheme entails

- pooling knowledge types used in models
- tracking models for documentation and historical reference of designs
- design history trace
- explicit representation of the artifact structure
- explicit representation of the knowledge structure, labels
- separation of data and knowledge from processing

In many problem domains, complex representational schemes are necessary due to the complexity of the artifacts being designed [?] and [?]. The inference engine, which uses the domain knowledge, also requires knowledge about how and when to apply that knowledge. The representation scheme includes

- models
- pools, or types, of knowledge
- procedural knowledge
- planning knowledge and
- meta knowledge.

Knowledge is grouped into categories, or pools, which contain all instances of a given category, such as equations or rules. Individual elements from pools of knowledge can be associated with each model. For example, in the design of a spacecraft subsystem, the same equation relating frequency to wavelength for electromagnetic radiation can be associated with the model for an infrared detector payload as well as with the communications subsystem model.

Building complex systems from specifications, which may be incomplete or even inconsistent, requires flexible, extensible models and the capability to utilize the knowledge associated with those models. A particular model and the knowledge associated with it may become obsolete as new data becomes available. New capabilities might be added to the system under design, requiring changes to the model. For example, in VEHICLES, if a circular orbit is selected the orbit model will automatically delete all information (equations, rules, and design configuration values) associated with elliptical orbits. Organizing the knowledge into 'pools of knowledge', such as a pool of equations or a pool of constraints, facilitates the addition of new knowledge categories as well as new instances within a given pool, or type, of knowledge.

To ensure consistency and maintain credibility we have chosen to explicitly represent the structure and associated knowledge of the artifact being designed, the models and knowledge about the design, and the representation scheme itself. Constraints and limitations about the models as part of the knowledge representation are also explicitly represented. For instance, each equation, in the pool of equations may have

- a source (person or reference),
- the time and date it was added,
- annotations (user readable),
- the assumptions upon which it is based and
- the conditions or limits for automatic validity checking.

A subset of equations (or rules, routines, tables, etc.) from the (appropriate) pool is selected for each model. The model itself may include information similar to that for the equations. This makes it possible to keep track of past designs, even when the models have changed. With this peripheral information, we can retrieve the exact model used at any point in a design. This is important for understanding why decisions were made. It might have been due to incompleteness of the models used. This also helps to identify how subsequent enhancements to the models may alter decisions and designs or what technology breakthroughs are needed to meet requirements. For example, in designing a phased array radar many transmit/receive modules are needed to provide sufficient power. A nominal value of 0.3 watts per module would take over 13,000 modules to supply 4000 watts. However, if we could only "afford" the weight of 8000 modules, we would need a technology improvement in modules for them to supply 0.5 watts (hopefully at the same weight).

It is possible to enhance, modify, restructure, remove or prune existing knowledge. Also, the models and associated knowledge can include redundant information in the form of different perspectives or different representations by linking associated concepts. Knowledge that is linked together cannot be automatically removed. An item which is replaced by an equivalent item will automatically have all links updated.

A useful feature in design is traceability. Tracing results makes it possible to identify how each result was derived. This design history provides the source of the data, which could be an equation, a rule, or a designer and the date the result was acquired. This automatically makes it possible to reconstruct the design path, the knowledge and the sequence of its application, which led to earlier design concepts, as well as to the current design concept.

Labels are assigned to the various procedures, facts and models. These labels are made as specific as possible to ensure that rich semantic content will be evident in the state of the program execution, or the state of the knowledge base. The traceability of results is facilitated when the data source is pgm_photon_flux or routine_weight_growth. It also makes the results more credible, because the user untrained in AI techniques can immediately see what factors (e.g. equations, routines) were used in the design.

Frequently, new models are incorporated that only partially overlap with the existing models. This is difficult since one model cannot simply be replaced with another. One type of knowledge that has been quite helpful in facilitating the combination of models is the explicit representation of the linked parameters, i.e. the parameters that are used in a particular model, but supplied from other models, routines, or tables. For instance, the total weight of the spacecraft relies on the individual subsystem weights. Replacement of a subsystem model must still provide the total weight of that subsystem.

Most of the integration of models is done by hand. As our systems continue to be used in additional design projects we are investigating what information would be needed to automate a greater portion of the process. Verifying the consistency and completeness of the knowledge is one aspect currently being worked on. It is already possible to identify conflicts when equations are being solved. A symbolic equation solver will flag problems when values are inconsistent.

The interface, between the design program and the knowledge base, can itself be modified if the knowledge base changes since it is just another type of knowledge. Thus, changes to the knowledge base are accompanied by changes to the interface portion of the knowledge base. The interface knowledge can be used to specify which portions of the updated knowledge base are applicable, thus allowing the proper knowledge from all domains to be utilized. This form of meta knowledge is necessary with a rapidly evolving model or knowledge base [?]. However, it makes the verification of knowledge base correctness, completeness and consistency much more difficult, since there is this meta knowledge which must also be verified since it is susceptible problems as the knowledge.

4 THE KNOWLEDGE HIERARCHY

Design knowledge, which addresses how to do design, what the design is and the requirements to be met, may be represented in a subtype/supertype hierarchy. KNOWMAN, used for synthesis of VLSI circuits, uses an object oriented database, which makes the hierarchical representation very natural. In VEHICLES, which uses a relational database, the hierarchical structure also is used. It provides a means to clearly represent the structure of the artifact being designed. It also makes the organization of complicated and voluminous knowledge and data much clearer and easier to access. Two subtypes of knowledge are declarative and procedural. Both of these contain many subtypes in a rich and tangled hierarchy.

Declarative knowledge includes state knowledge, parameterized knowledge, knowledge about goals and assertions. Most statements of facts, descriptions of relationships between objects, and characteristics of objects are members of this type.

State knowledge is knowledge about all of the aspects of the design space: past, present and potential. This is a complex area of knowledge since there are so many aspects that need to be considered. We have selected several candidate areas for inclusion in this type of knowledge:

design space	state history
design paths	current design state
state-dependent data	design approaches
design domains	design drivers
design styles	design situations
design evaluation	design decisions

Parameterized knowledge is particularly important in MBR since the data contained is used to fine tune the models. Equations, tables, ordered lists, graphic knowledge, definitions and image processing data are all examples of the types of knowledge included in this category. Also included are the ranges and units of the attributes associated with the model.

The hierarchy based on the goals associated with the specification being used to drive the planner includes constraints, goals, lower and upper bounds, best achievable designs and actual designs.

Our work in knowledge classification and interaction is ongoing. Other types of knowledge spans many levels, from primitives to higher level constructs. The classification scheme is not rigid, so that the groupings may vary depending on the context. These categories include:

- Symbols, operators, numerics, strings Equations, expressions, functions
- Tables, lists, sets
- Rules, constraints, requirements, goals, scenarios
- Definitions, annotations Instances, data, constants, enumerated options
- Models, frames, scripts
- Graphics, image processing (postscript, giff, pic)
- Report styles, graph styles, highlight items

Under the domain of procedural knowledge, we see the subtypes of heuristics, algorithms, operators, reasoning, rule usage and planning. In KNOWMAN the procedural knowledge is stored in the form of hierarchical data flow graphs where specific procedures are associated with the nodes, and values flow between the nodes. Declarative knowledge can be associated with the procedures by binding specific Prolog code to a given node based on the current state of the design.

5 SUMMARY AND STATUS

We have presented a representational scheme that makes use of models based on pools of knowledge. The structure of the knowledge is semantically rich. This is apparent in the wealth of types, or pools, and the many levels of knowledge that can be represented. Explicit representation will facilitate checking the consistency and ensuring appropriate applicability of information in the knowledge base.

The separation of the inference engine from the knowledge base makes possible the storage of the data in a form that is easily extensible, easily reorganizable and extremely flexible. It also makes it possible to have work proceed on both in parallel.

Complex interactions between subsystems are handled by explicit representation. This facilitates managing the evolution of the models, but it also makes it evident that there is a need for a consistency checker for the knowledge base to ensure the completeness and consistency of the knowledge.

The non-monotonic nature of the evolving models necessitates including *what* and *when* changes occur. This makes it possible to capture design history, providing traceability, and makes the model state dependence explicit. It also helps maintain credibility with the user.

The methodology described here has been, to a large extent, implemented. We have followed the philosophy of separating the knowledge from the inference engine. This became a necessity as the representation scheme frequently changed as new cpaabilities are continually being incorporated. This

There are some caveats which should be mentioned. As the generality and power of the MBR system grows the performance, especially in terms of speed, decreases. There is also considerable overhead incurred by checking for validity and consistency. The scope of the knowledge represented, which includes both the knowledge itself and its applicability conditions, requires more storage space and necessitates more work in incorporating new models or system capabilities. We are currently investigating solutions to these problems, and see great potential in MBR.

References

[1] K. Bellman and A. Gillam, "A knowledge-based approach to the conceptual design of space systems," in *Proceedings of the 1988 Eastern MultiConference*, pp. 23–27, Mar. 1988.

[2] A. Gillam, "A knowledge-based approach to planning the design of space systems," in *Proceedings of the 1989 Eastern MultiConference*, 1989.

[3] H. Afsarmanexh, *The 3DIS, An Extensible Object Oriented Framework for Information Management.* PhD thesis, USC, 1985.

[4] D. Knapp, *A Planning Model of the Design Process.* PhD thesis, USC, 1986.

[5] D. Knapp and A. Parker, "A unified representation for design information," in *Proceedings of the 1985 Conference on Hardware Description Languages, IFIP*, 1985.

[6] J. Doyle, "A truth maintenance system," *AI Journal*, vol. 12, no. 3, 1979.

[7] E. Shortliffe, *Computer-Based Medical Consultations: MYCIN.* Elsevier, NY, 1976.

[8] R. Davis, "Applications of meta-level knowledge to the construction, maintenance, and use of large knowledge bases," in *HPP Memo and AI Memo*, (Stanford University), 1976.